LISTING OF CLAIMS:

1-85. (Cancelled)

86. (Currently amended) ~~The method of claim 85, wherein the step of making the runtime value of the silent guard dependent on the runtime value of the program variable, comprises~~ A computer implemented method for adding tamper resistance to a software program, the method comprising:

selecting a program block that computes a result necessary for proper execution of the software program, the program block comprising at least one program instruction;

selecting a silent guard for the program block;

selecting an insertion point in the software program;

selecting a program variable in the software program;

determining the expected value of the program variable at the insertion point;

determining the expected value of the silent guard at the start of execution of the program block;

installing a second computation that includes the runtime value of the program variable, such that the result of the second computation is corrupted if the runtime value of the program variable is not equal to the expected value of the program variable at the insertion point; ~~and~~

setting the silent guard equal to the result of the second computation, such that the runtime value of the silent guard equals the expected value of the silent guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point; and

installing a first computation dependent on the silent guard in the software program, such that if the runtime value of the silent guard is not equal to the expected value of the silent guard then the first computation causes the result computed by the program block to evaluate improperly, causing the software program to execute improperly.

87. (Currently amended) The method of claim 86, wherein the ~~silent guard~~ second computation uses both the expected value of the program variable and the runtime value of the program variable.

88-93. (Cancelled)

94. (Previously presented) A recordable computer media having a tamper resistant software program recorded thereon, the tamper resistant software program comprising~~,~~:

a program variable having an expected value at a first dependency point in the software program and an expected value at a second dependency point in the software program, the expected value at the first dependency point not being equal to the expected value at the second dependency point;

a silent guard variable having an expected value at the first dependency point;

a mathematical computation that includes the runtime value of the silent guard variable and an expected term, the expected term being set based on the expected value of the silent guard variable at the first dependency point;

a supplementary silent guard variable having an expected value at the second dependency point in the software program;

wherein the runtime value of the program variable is dependent on the result of the mathematical computation which is dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable will not equal the expected value of the program variable at the first dependency point if the runtime value of the silent guard variable does not equal the expected value of the silent guard variable at the first dependency point, which will cause the software program to execute improperly; and

wherein the runtime value of the program variable at the second dependency point is dependent on the runtime value of the supplementary silent guard variable, such that the runtime value of the program variable will not equal the expected value of the program variable at the second dependency point if the runtime value of the supplementary silent guard variable does not equal the expected value of the supplementary silent guard variable at the second dependency point, which will cause the software program to execute improperly.

95. (Previously presented) A recordable computer media having a tamper resistant software program recorded thereon, the tamper resistant software program comprising:

a program variable having an expected value at a first dependency point in the software program and an expected value at a second dependency point in the software program, the expected value at the first dependency point not being equal to the expected value at the second dependency point;

a silent guard variable having an expected value at the first dependency point and an expected value at the second dependency point in the software program;

a mathematical computation that includes the runtime value of the silent guard variable and an expected term, the expected term being set based on the expected value of the silent guard variable at the first dependency point;

wherein the runtime value of the program variable is dependent on the result of the mathematical computation which is dependent on the runtime value of the silent guard variable, such that the runtime value of the program variable will not equal the expected value of the program variable at the first dependency point if the runtime value of the silent guard variable does not equal the expected value of the silent guard variable at the first dependency point, which will cause the software program to execute improperly; and

wherein the runtime value of the program variable at the second dependency point is dependent on the runtime value of the silent guard variable at the second dependency point, such that the runtime value of the program variable will not equal the expected value of the program variable at the second dependency point if the runtime value of the silent guard variable does not equal the expected value of the silent guard variable at the second dependency point, which will cause the software program to execute improperly.

96-102. (Cancelled)

103. (Currently amended) The method of claim ~~85~~ 86, wherein the insertion point is separated from the installation point of the ~~branch instruction~~ first computation by a plurality of program instructions of the software program.

104. (Previously presented) The recordable computer media of claim 94, wherein the first dependency point is separated from the second dependency point by a plurality of program instructions of the software program.

105. (Previously presented) The recordable computer media of claim 95, wherein the first dependency point is separated from the second dependency point by a plurality of program instructions of the software program.

106-119. (Cancelled)

120. (New) A recordable computer media having a tamper resistant software program recorded thereon, the tamper resistant software program comprising:

a program block that computes a result necessary for proper execution of the software program, the program block comprising at least one program instruction;

a silent guard for the program block, the silent guard having an expected value at the start of execution of the program block;

a program variable, the program variable having an expected value at an insertion point in the software program;

a second computation that includes the runtime value of the program variable, such that the result of the second computation is corrupted if the runtime value of the program variable is not equal to the expected value of the program variable at the insertion point; the silent guard being set equal to the result of the second computation, such that the runtime value of the silent guard equals the expected value of the silent

guard if the runtime value of the program variable equals the expected value of the program variable at the insertion point;

a first computation dependent on the silent guard, such that if the runtime value of the silent guard is not equal to the expected value of the silent guard then the first computation causes the result computed by the program block to evaluate improperly, causing the software program to execute improperly.

121. (New) The recordable computer media of claim 120, wherein the second computation uses both the expected value of the program variable and the runtime value of the program variable.

122. (New) The recordable computer media of claim 120, wherein the insertion point is separated from the first computation by a plurality of program instructions of the software program.

.